UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Seguimiento de objetos mediante constricción espacial de puntos TILDE.

Autor: César Augusto Betancur Cruz.
Tutor: Marcos Escudero Viñolo.
Ponente: Jesús Bescós Cano.

Junio 2016

Seguimiento de objetos mediante constricción espacial de puntos TILDE.

Autor: César Augusto Betancur Cruz
Tutor: Marcos Escudero Viñolo
Ponente: Jesús Bescós Cano

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2016

Trabajo parcialmente financiado por el Ministerio de Economía y Competitividad del Gobierno de España bajo el proyecto TEC2014-53176-R (HAVideo) (2015-2017)

Resumen

El objetivo de este trabajo fin de grado es el diseño y desarrollo de un algoritmo de seguimiento de objetos basado en puntos de interés. En particular, se busca evaluar las potenciales mejoras que supone introducir un esquema de detección de puntos de interés recientemente publicado que ofrece sustanciales mejoras frente a otros métodos más populares.

Para el diseño del algoritmo, primero se realizará un estudio de los algoritmos de seguimiento de objetos existentes. Dentro de este análisis se va a profundizar en los algoritmos basados en la detección de puntos de interés. Para facilitar su comprensión y potencial mejora, el diseño del algoritmo realizado se adapta al modelo básico de algoritmo de seguimiento mediante detección de puntos de interés.

Del estudio realizado, se propone utilizar el detector TILDE (Temporally Invariant Learned Detector). TILDE es un detector de puntos de interés basado en entrenamiento que proporciona alta estabilidad a cambios de iluminación y apariencia. Las correspondencias entre puntos TILDE se usarán para realizar constricciones espaciales de la posición del objeto seguido entre cuadros consecutivos. El resultado se refina utilizando un esquema clásico de correlación cruzada.

Finalmente se evalúa el algoritmo desarrollado frente a un conjunto de evaluación estándar. A partir de esta evaluación, se discuten sus puntos fuertes y débiles a fin de determinar su usabilidad y sus usos potenciales. Adicionalmente, se realiza una evaluación comparativa con otros algoritmos de seguimiento, a fin de poder cuantificar su rendimiento relativo a otros métodos del estado del arte.

Los resultados experimentales validan parcialmente el diseño y desarrollo del algoritmo, y sugieren que el uso de los puntos TILDE puede generar estrategias de constricción espacial de utilidad para algoritmos basados en correlación cruzada.

Palabras clave

Seguimiento, puntos de interés, detección de puntos, descripción, búsqueda de correspondencias.

Abstract

The aim of this work is the design and development of a video object tracker based on point-of-interest (PoI). Specifically, we focus on the potential benefits of including a recently published PoI detector in the core of the tracking process.

The design of the algorithm starts by studying existing tracking algorithm emphasizing in methods using PoI in any stage of their tracking algorithm. In order to provide a flexible framework on which to develop potential improvements, the algorithm's design builds on the basic PoI-based tracking scheme.

From the state-of-the-art, we propose to use TILDE (Temporally Invariant Learned Detector) as the PoI detection method. TILDE is a train-based PoI detection method which is claimed to provide stable results to change in illumination and appearance. TILDE-driven correspondences are used to spatially constrain the target position between consecutive frames. Final result is refined by means of a classic cross-correlation method.

The tracker is experimentally evaluated in a generic evaluation corpus. From this evaluation we aim to discuss on their benefits and drawbacks. Furthermore, we also compare the tracker against state-of-the-art trackers, in order to quantitatively contextualise its operation.

Experimental results partially validate the design and development of the algorithm and suggest that the use of TILDE may help to generate robust constraining schemes for trackers based on cross-correlation.

Keywords

Tracking, points of interest, point detection, description, matching.

Agradecimientos

Este trabajo representa el final de una de las etapas más emocionantes de mi vida. No ha sido poco el esfuerzo invertido pero tampoco lo ha sido la ayuda que se me ha brindado. Así, solo cabe dar las gracias a los que me han acompañado en este viaje.

En primer lugar a mis compañeros de carrera, especialmente a Esparza, Rubén y Adri, por su ayuda incondicional en los momentos difíciles y su sentido del humor en los no tan difíciles.

A los profesores, por las lecciones, la motivación pero sobre todo por su compresión. Hay ingenieros bondadosos, y yo, espero llegar a ser ambas cosas algún día.

A la Universidad Autónoma de Madrid por darme la posibilidad de realizar un curso de intercambio inolvidable en el que encontré nieve, renos y algo más.

A mi tutor Marcos, a quien ya considero un amigo, por ofrecerme la oportunidad de hacer este TFG con él, y más aun, por ayudarme a llevarlo a buen puerto. Gracias por la amabilidad con la que me has tratado.

Y por último a mis padres, por el apoyo y cariño incalculable que me han dado cada día en los pequeños y en los grandes detalles. Gracias por la confianza que habéis depositado en mi, este logro es tan vuestro como mío.

César Augusto Betancur Cruz. Junio 2016

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Los seres vivos necesitan de métodos para obtener información de su entorno a fin de poder interactuar con el. Además, los seres dotados de un sistema de visión son especialmente eficaces en esta tarea ya que a través de este reciben hasta el 80 % de la información de su entorno. Entre las diversas utilidades para las que se puede aprovechar esta información se encuentra el seguimiento de objetos. Gracias a esta habilidad es posible reaccionar en consecuencia para realizar multitud de tareas básicas.

El vertiginoso desarrollo de las nuevas tecnologías ha hecho posible transmitir esta capacidad entre otras a sistemas artificiales. Por otro lado, el seguimiento de objetos representa uno de los objetivos principales en el campo de la visión por computador debido a sus potenciales aplicaciones. Sin embargo, esta capacidad natural de los seres vivos resulta bastante costosa de implementar en sistemas artificiales.

Numerosos métodos han sido ideados buscando aprovechar la elevada capacidad de calculo de la que disponen los computadores modernos. Estos métodos se descomponen en una serie de algoritmos encargados de llevar a cabo procesos mas sencillos como caracterizar el objeto, predecir su movimiento o localizarlo en momento.

Entre estos métodos se encuentran los basados en detección de puntos de interés (PoIs). Los detectores de PoI son algoritmos encargados de identificar características singulares en una imagen a fin de adquirir cierta información de contexto.

Este tipo de métodos exigen un detector de PoIs fiable y robusto de cara a afrontar las dificultades habituales que se presentan en las secuencias de video. Existen numerosas opciones que ofrecen resultados satisfactorios sin embargo, la mayoría muestran un considerable descenso en el rendimiento ante cambios ambientales.

Figura 1.1: Secuencia **bolt1** extraída del *dataset* VOT2016. A la izquierda se indica la posición y tamaño del objetivo en el primer cuadro de la secuencia. A la derecha el resultado del seguimiento.

TILDE es un algoritmo de detección de PoIs recientemente desarrollado y cuya motivación es la de subsanar estas deficiencias. Más concretamente, TILDE ha sido diseñado para ofrecer una alta estabilidad bajo bruscos cambios meteorológicos y de iluminación.

La motivación principal de este trabajo es implementar un sistema de seguimiento de objetos que aproveche las características ofrecidas por TILDE, ademas de determinar su la viabilidad y rendimiento sometiéndolo a una evaluación y comparándolo con otros algoritmos.

1.2. Objetivos

El propósito principal de este trabajo es el diseño e implementación de un algoritmo de seguimiento de objetos que cuente con las ventajas que ofrece TILDE. Se requiere por tanto conocer los métodos existentes así como sus características a fin de determinar una solución robusta. Para llevar a cabo este trabajo, se consideran los siguientes objetivos:

1. **Estudio detallado del estado del arte**. Buscar estrategias no exploradas para el seguimiento de objetos basado en puntos de interés.

2. **Desarrollo de un algoritmo de seguimiento.** Este algoritmo debe aprovechar la estabilidad en la detección de puntos de interés ofrecida por el algoritmo TILDE.

3. **Evaluación de la solución propuesta**. Determinar el rendimiento del algoritmo con respecto al estado del arte actual.

4. **Generación de conclusiones**. Identificar puntos fuertes y débiles del algoritmo desarrollado y evaluar la consecución de los objetivos propuestos.

1.3. Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1. Motivación, objetivos del proyecto y estructura de la memoria.
- Capítulo 2. Estudio del estado del arte de los métodos de seguimiento y de detección de PoIs. Explicación de las bases teóricas sobre las que se apoyan.
- Capítulo 3. Descripción de los métodos elegidos para la implementación del algoritmo.
- Capítulo 4. Exposición de los experimentos realizados y de los resultados obtenidos.
- Capítulo 5. Extracción de conclusiones. Enumeración de las posibles mejoras y lineas de trabajo futuro.
- Bibliografía.

Capítulo 2

Estado del arte

2.1. Introducción

El seguimiento de objetos por métodos automáticos de visión artificial (en adelante *tracking*) es el análisis de secuencias de video con el propósito de establecer la localización de un objetivo sobre cada cuadro de una secuencia [1]. El *tracking* representa ademas uno de los problemas más desafiantes en el campo de la visión por computador. Esto se debe a la enorme diversidad de retos y escenarios posibles que pueden aparecer en situaciones realistas. Estas situaciones adversas pueden ser causadas por características físicas del objeto, o darse debido a la propia naturaleza de la secuencia. El diseño de herramientas y métodos capaces de hacer frente a estos retos, desemboca habitualmente en algoritmos complejos muchas veces dominados por soluciones heurísticas.

A pesar de su alta complejidad el *tracking* es uno los campos con mas futuro en investigación debido a la enorme diversidad de aplicaciones prácticas que ofrece como pueden ser video vigilancia, vehículos auto-pilotados o prevención de accidentes. En este capitulo se realiza un análisis de los métodos más utilizados para el *tracking,* señalando sus características y los problemas que deben afrontar. Posteriormente se incluye una introducción a la detección y descripción de puntos de interés en imágenes, proceso que representa uno de los pilares principales sobre los que se apoya el algoritmo propuesto en este trabajo fin de grado (en adelante TFG). Finalmente se presenta el esquema general en el que se basan los algoritmos de *tracking* mediante detección y descripción de PoIs.

2.2. Seguimiento de objetos

2.2.1. Tipos de algoritmos de seguimiento

- ***Matching***[1]. Este grupo de *trackers*[2] es uno de los más comunes y se basa en la realización de una correspondencia para cada par de cuadros consecutivos con el fin de hallar la nueva posición del objeto.

- ***Matching* con modelo de apariencia**. Este grupo de métodos introduce una mejora sobre el anterior. En este caso el sistema mantiene un modelo de apariencia del objeto aprendido durante los cuadros anteriores. Para la correspondencia se utiliza entonces tanto el objeto en el cuadro anterior como este modelo de apariencia. De este modo el seguimiento del objeto es más fiable en caso de que éste haya sufrido cambios en su apariencia.

- **Clasificación discriminativa**. Se basa en realizar un clasificación entre el primer plano (el objetivo) y el fondo. De esta manera se sigue al objetivo discriminado la información correspondiente al fondo. es decir, se sigue al objeto como lo complementario al fondo.

- **Clasificación discriminativa y *matching* con constricciones**. Estos métodos se basan en los anteriores y son en general más complejos debido a que tratan de identificar el objeto con la mínima información posible para tener una menor dependencia de su apariencia inicial. Son por tanto técnicas utilizadas para secuencias mas específicas como aquellas en las que la perspectiva o apariencia del objeto cambian drásticamente.

2.2.2. Características de los algoritmos

- **Región objetivo**. Uno de los pasos críticos en el proceso de *tracking* tiene lugar al inicializar el algoritmo. Esta fase suministra al sistema toda la información posible sobre el objeto a seguir.. Los *trackers* de propósito general solicitan como parámetro de entrada una región de la imagen en la que poder identificar el objetivo. Esta región conocida como *bounding box*, se actualiza en cada cuadro como resultado del seguimiento indicando así la posición y tamaño del objetivo. Algunos de los detractores de este nivel de seguimiento argumentan que debido a su imprecisión se incluyen de forma involuntaria regiones del fondo que pueden acabar siendo confundidas con el objeto [1]. Existen alternativas que utilizan

[1] Correspondencia
[2] Algoritmos de seguimiento

elipses o incluso el propio contorno del objeto aunque son más comunes en aplicaciones especificas como imágenes medicas. Otros *trackers* se basan en el seguimiento de *blobs*, regiones o super-píxeles, que son zonas cuyos pixeles que presentan características comunes (color, movimiento, etc).

- **Representación de la apariencia**. Para la representación de la apariencia del objeto existen tres métodos básicos ampliamente extendidos. El más simple de estos métodos es una matriz con los valores de brillo del objeto de tal manera que la representación conforma un imagen en si misma. Por otra parte tenemos la representación mediante histograma mediante la cual se preserva la distribución estadística de los colores de la imagen pero libera totalmente cualquier restricción respecto a su distribución espacial proporcionando de esta forma máxima flexibilidad al objeto en su movimiento. Por último, el vector de características sirve para codificar la apariencia geométrica del objeto.

- **Estimación del movimiento**. Por lo general, la mayoría de *trackers* asumen que el objetivo se va a encontrar en un área próxima a su localización previa. Por ello, la búsqueda se realiza en torno al área espacial que ocupa el objetivo en el cuadro anterior siendo todas las direcciones de desplazamiento igual de probables. Este método ofrece en general buenos resultado ya que permite adaptarse a cambios de dirección del movimiento del objeto, aunque resulta insuficiente cuando el movimiento es muy rápido. Para solventar esta limitación se usan otras técnicas en las que la búsqueda se centra en zonas estimadas mediante predicción en base al movimiento previo del objeto.

- **Método de seguimiento**. El núcleo de un algoritmo de seguimiento es el método utilizado para calcular la localización mas probable del objeto. Estos métodos, expuestos en la sección 2.2.1, tienen como finalidad estimar la nueva posición del objeto a través de los cuadros de la secuencia.

- **Modelo de actualización**. Mediante la actualización de la apariencia, la búsqueda del objetivo en el nuevo cuadro se realiza en base a las características del modelo de apariencia en su última posición detectada. Esto permite que la apariencia del objeto evolucione a lo largo de la secuencia . Sin embargo algunos algoritmo no realizan esta actualización de tal manera que se mantienen el mismo modelo para la búsqueda durante toda la secuencia. Otras alternativas utilizan un sistema de actualización mediante predicción de la apariencia pensados para secuencias largas.

2.2.3. Retos y dificultades

El seguimiento de objetos en secuencias de video presenta una amplia gama de situaciones con dificultades asociadas que deben ser tenidas en cuenta. Estas dificultades son producto de la naturaleza de la secuencia o de la limitaciones en las técnicas [2]. Las más comunes son:

- **Cambios de iluminación**. Un cambio de iluminación durante la secuencia altera los valores de los pixeles de la imagen. Métodos basados en la obtención de características en función de estos valores, como calculo de histogramas, son poco efectivos ante este problema. Detectores de características robustos a estos cambios, como por ejemplo TILDE [3], son potencialmente útiles.

- **Oclusión**. Se denomina oclusión al hecho de que el objetivo se vea total a parcialmente cubierto por la acción del entorno o de otro objeto. Los sistemas que guardan un modelo de apariencia son los más eficaces ante esta situación.

- **Deformación**. Las características físicas de los objetos determinan la forma y el grado de las deformaciones que pueden experimentar. Clasificando los objetos según estas características se pueden distinguir tres tipos principales: objetos rígidos, objetos flexibles y objetos articulados, siendo estos dos últimos los que presentan mayores dificultades.

- **Ruido**. El ruido es una característica común de las señales digitales por la cual los valores de sus muestras se ven modificados de manera aleatoria dificultando así su procesamiento. Este ruido proviene habitualmente del proceso de captación de la señal aunque también puede aparecer en las etapas de codificación y cuantificación o transmisión. Numerosos métodos han sido desarrollados para combatir este problema entre los que cabe destacar el suavizado mediante filtrado debido su simplicidad y efectividad.

- **Cambios de perspectiva**. Además de los cambios de posición, los objetos también son susceptibles de experimentar rotaciones. Es en este caso cuando un objeto puedo cambiar radicalmente su apariencia dificultando enormemente su seguimiento. Para solventar este problema existen métodos de corrección de perspectiva mediante homografías que serán discutidos en la sección 2.4.5.

- **Emborronado**. Debido a la limitada resolución temporal de los dispositivos de captación es habitual que se produzca un emborronado en los cuadros de una secuencia cuyos objetos se mueven muy rápidamente. Este defecto aunque puede ser corregido mediante técnicas de procesado de imagen, es habitual

afrontarlo utilizando detectores de características robustos como en el caso de los cambios de iluminación.

2.3. Detección de puntos de interés

La detección de puntos de interés (PoI) es un problema ampliamente estudiado debido a que sirve como base para una gran variedad de algoritmos de visión por computador. Diversos métodos han sido desarrollados tratando de mejorar el rendimiento en función de los distintos escenarios posibles. Estas aproximaciones se basan en detectar puntos singulares o representativos en una imagen en función de sus características. Dichos puntos singulares son lo que se conoce como PoI y se detectan aplicando diferentes criterios como son su posición en la imagen, el valor de los pixeles adyacentes o su estabilidad ante distintas perturbaciones. Tras el proceso de detección habitualmente se utilizan algoritmos conocidos como descriptores cuya finalidad es la de definir los puntos de manera unívoca para su posterior procesamiento. En esta sección se van a presentar algunos de los algoritmos más usados en la detección de PoI y las bases teóricas sobre las que se apoyan.

2.3.1. Espacio Escala

El espacio escala es una representación estructural de la imagen formada por el conjunto de imágenes obtenidas al aplicar sucesivamente un suavizado constante llamado factor de escala. Este suavizado se realiza típicamente aplicando un filtro Gaussiano con el fin no producir nuevas estructuras en la imagen y se define mediante la ecuación 2.1:

$$g(x, y; t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/2t} \tag{2.1}$$

Sea una imagen $\psi(x, y)$, su representación en el espacio escala se expresa mediante la ecuación 2.2

$$L(x, y; t) = g(x, y; t) * \psi(x, y) \tag{2.2}$$

, donde $*$ denota la operación de convolución. De esta forma el resultado se puede considerar como un imagen en 3 dimensiones: x, y, t en la que la nueva dimensión t representa la escala, es decir las veces que se le ha aplicado el filtrado Guassiano. El primer nivel del espacio escala es la imagen original y se corresponde con $t = 0$ tal y como muestra la ecuación 2.3

$$L(x, y; 0) = g(x, y; 0) * \psi(x, y) = \psi(x, y) \tag{2.3}$$

Esta representación es especialmente útil a la hora de buscar PoI ya que permite obtener información acerca de la estabilidad de cierta zona o región de la imagen.

2.3.2. Detectores en el espacio escala

La representación espacio escala una herramienta ampliamente utilizada en diversos algoritmos de detección de PoI ya que proporciona información acerca de la estabilidad de una determinada zona de la imagen a diferentes niveles de perturbación. En esa sección se van a introducir algunos de los algoritmos de detección de PoI mas utilizados que se apoyan en la representación espacio escala [4].

- **Diferencia de Gaussianas (DoG)**. El DoG consiste en la resta de dos imágenes consecutivas del espacio escala de tal forma que el resultado se puede expresar como muestra la ecuación 2.4:

$$D(x, y; t) = L(x, y; t_{i+1}) - L(x, y; t_i) \tag{2.4}$$

  El *kernel*[3] Gaussiano utilizado en el espacio escala tiene el efecto de filtro paso bajo puesto que afecta mayoritariamente a las frecuencias altas en la imagen. Al restar dos de estas imágenes suavizadas se obtiene una imagen que mantiene solo el rango de frecuencias comprendidas entre ellas de tal manera que se puede considerar que el DoG actúa de manera análoga a una filtro paso banda.

- **Laplaciano de la Gaussiana (LoG)**. Consiste en aplicar el operador Laplaciano a la representación espacio escala de una imagen y se define mediante la ecuación 2.5:

$$\nabla^2 L(x, y; t) = \frac{\partial^2 L(x, y; t)}{\partial^2 x} + \frac{\partial^2 L(x, y; t)}{\partial^2 y} = L_{xx} + L_{yy} \tag{2.5}$$

  El LoG arroja fuertes valores negativos en presencia de *blobs* claros y fuertes valores positivos en presencia de *blobs* oscuros de radio $2\sqrt{t}$ en ambos casos. El principal problema de este operador es que su respuesta decrece conforme aumenta la escala. Para solucionar este problema se utiliza la versión normalizada del LoG, $\nabla^2_{norm} L = t(L_{xx} + L_{yy})$ que permite detectar *blobs* a diferentes escalas.

- **Determinante del Hessiano (DoG)**. La matriz Hessiana de una función es una matriz cuadrada que contiene las segundas derivadas parciales de di-

[3] Filtro

cha función. Llamamos Hessiano al determinante de esta matriz y se expresa mediante la ecuación 2.6:

$$HL(x,y;t) = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{bmatrix} \qquad \det HL(x,y;t) = L_{xx}L_{yy} - L_{xy}^2 \tag{2.6}$$

La principal característica de este determinante es que describe la curvatura de la función y por lo tanto ofrece información relevante para identificar puntos de interés. Se considera que la matriz Hessiana es definida positiva si son positivos sus autovalores y definida negativa en el caso contrario. Por otra parte, un punto crítico o punto estacionario es aquel punto de la imagen en el que el gradiente es nulo. Si la matriz Hessiana es definida positiva en un punto estacionario indica la presencia de un mínimo local mientras que si es definida negativa se trata de un máximo local. En caso de ser los autovalores mixtos se trata de un punto de silla. Finalmente al calcular el determinante del Hessiano se puede extraer también parte de esta información en función del signo del resultado de tal que manera que si es positivo de trata de un punto máximo o mínimo mientras que si el resultado del DoH es negativo indica la presencia de un punto de silla.

2.3.3. Scale-Invariant Feature Transform (SIFT)

SIFT [5] es una algoritmo de detección y descripción de PoI basado en la aplicación del operador DoG sobre una versión piramidal del espacio escala. La parte de detección de este algoritmo se descompone en cinco etapas principales:

- **Generación de un espacio escala híbrido**. Se comienza obteniendo una representación híbrida en la que se calcula tanto el espacio escala de la imagen original como el de sucesivas versiones diezmadas basándose en el hecho de que el error cometido al sub-muestrear un imagen es menor cuanto mayor es la escala. Se denomina octava a cada nivel de esta pirámide que a su vez están formadas por $s+3$ escalas. La primera octava es el espacio escala de la imagen original. El primer nivel de las sucesivas octavas se obtiene diezmando la imagen de la octava anterior cuya escala es$\sigma' = 2\sigma$.

- **Aplicación del DoG**. Se aplica el algoritmo DoG a cada octava generando de esta forma $s+2$ imágenes por octava. Posteriormente se realiza una primera selección de puntos comparando cada pixel de cada imagen DoG con sus 26 vecinos, los 8 adyacentes que pertenecen a la misma escala más los 9 vecinos de cada escala DoG anterior y posterior. Se considera punto de interés si el punto analizado es un máximo o un mínimo local en esta comparación.

- **Localización precisa y selección de puntos**. El siguiente paso consiste en localizar los puntos de manera precisa con el fin de mejorar la estabilidad en las etapas posteriores, para ello se calcula el valor de $D(\mathrm{x}) = D(x, y; t)$ en un pequeño desplazamiento $\triangle\mathrm{x}$ en torno al punto candidato obtenido en el paso anterior utilizando la serie de Taylor presentada en ecuación 2.7:

$$D(\triangle\mathrm{x}) = D(\mathrm{x}) + \frac{\partial D^T(\mathrm{x})}{\partial \mathrm{x}}\triangle\mathrm{x} + \frac{1}{2}\triangle\mathrm{x}^T\frac{\partial^2 D(\mathrm{x})}{\partial \mathrm{x}^2}\triangle\mathrm{x} \tag{2.7}$$

  Lo que se busca es el mínimo valor de desplazamiento$\triangle\hat{\mathrm{x}} = (\triangle\hat{x}, \triangle\hat{y})^T$ igualando a cero su primera derivada. Si ambas componentes del puntos son menores que 0.5 se asume que la posición precisa de dicho punto es $\hat{\mathrm{x}} = \mathrm{x} + \triangle\hat{\mathrm{x}}$, en caso contrario se repite el proceso entorno al vecino correspondiente.

- **Descarte de puntos poco contrastados**. Sea un punto candidato detectado siguiendo los pasos anteriores $\hat{\mathrm{x}} = \mathrm{x} + \triangle\hat{\mathrm{x}}$, consideramos que es un punto poco contrastado si $|D(\triangle\hat{\mathrm{x}})| \leq 0{,}03$.

- **Descarte de puntos localizados en bordes**. Uno de los problemas del DoG es que arroja una fuerte respuesta a lo largo de bordes en la imagen. Para solucionarlo se analiza la curvatura del DoG calculando su matriz Hessiana $HD(\hat{\mathrm{x}})$ en cada punto detectado. Una relación entre sus autovalores $r = \lambda_1/\lambda_2$ mayor que 10 indica una curvatura con dirección predominante por lo que se descarta dicho punto.

2.3.4. El descriptor SIFT

Una vez obtenidos los puntos de la interés de una imagen el siguiente paso es identificarlos de manera unívoca con el fin de hacerlos robustos a rotaciones, cambios de posición o de iluminación. Para este fin se utilizan algoritmos conocidos como descriptores. SIFT cuenta con un algoritmos de descripción basado en el reconocimiento de distribuciones del gradiente. Sus etapas principales son:

- **Asignación de una orientación principal a cada punto**. Se obtiene el gradiente de la imagen $L(x, y; t)$ sobre la misma escala t en la que se detecto el PoI. Posteriormente se genera un histograma de orientaciones de 36 columnas mediante votación de los pixeles entorno al PoI. Para la votación se pondera el modulo del gradiente por una Gaussiana centrada en el PoI de $\sigma' = 1{,}5\sigma$. La orientación principal se obtiene tomando el pico principal del histograma y cualquier otro pico que represente al menos el 80 % del principal, de este modo

un mismo PoI puede tener más de una orientación principal por lo que en este proceso los PoI pueden replicarse.

- **Distribución de las orientaciones del gradiente**. Se comienza calculando la magnitud y orientación del gradiente en una región de 16x16 pixeles en torno al PoI. La orientación se rota con respecto de la orientación principal calculada en el paso anterior mientras que la magnitud se pondera con un Gaussiana. De cada subregión de 4x4 pixeles se obtiene un histograma de orientaciones, esta vez de 8 columnas. Estas orientaciones se escalan por la magnitud del gradiente. Finalmente se concatenan estos 16 histogramas de 8 columnas dando lugar a un vector de 128 valores que será finalmente el descriptor del punto.

2.3.5. Temporally Invariant Learned DEtector (TILDE)

TILDE [3] es un algoritmo de detección de PoI cuya principal ventaja es la gran estabilidad que ofrece frente a cambios de meteorológicos y de iluminación. Esta característica ha sido siempre uno de los puntos débiles de la mayoría de algoritmos de detección. Por su parte, TILDE es capaz ofrecer resultados robustos entre escenas que han sido tomadas con grandes diferencias de iluminación o bajo condiciones meteorológicas adversas en el caso de localizaciones exteriores.

TILDE se basa en el aprendizaje mediante regresión proporcionando un conjunto de imágenes en las que ademas se ya se han identificado puntos potencialmente estables. Estas imágenes de entrenamiento han de ser de la misma escena y capturadas desde el mismo punto de vista pero bajo distintas condiciones meteorológicas y de iluminación . Para obtener el set de entrenamiento se aplica un detector de puntos como por ejemplo SIFT sobre el conjunto de imágenes antes mencionado. Se marcan como muestras positivas aquellos puntos que se repiten a lo largo de la mayoría de la pila de imágenes. Por el contrario, aquellos puntos no detectados o detectados aisladamente son marcados como muestras negativas. Cabe destacar que la regresión se realiza procurando se que devuelvan puntuación similares para los mismos puntos sobre el set de entrenamiento adquiriendo de esta forma mayor consistencia a la hora de identificar puntos bajo distintas condiciones.

Tras el entrenamiento, el algoritmo es capaz de proporcionar un mapa de puntuaciones para imágenes nuevas en el que los valores más altos corresponden a muestras positivas mientras que a las muestras negativas se les asigna un valor lo mas pequeño posible con el fin de ser fácilmente descartables. De esta forma los PoI se obtienen haciendo una supresión de no máximos en el mapa de puntuaciones. La regresión se expresa mediante la ecuación 2.8:

$$\mathbf{F}(\mathbf{x};\omega) = \sum_{n=1}^{N} \delta_n \max_{m=1}^{M} \mathbf{w}_{nm}^{\mathsf{T}}\mathbf{x} \tag{2.8}$$

, donde $\mathbf{x}$ son una serie de característica extraídas de la imagen y $\omega = [\mathbf{w}_{11}^{\mathsf{T}}, \ldots, \mathbf{w}_{MN}^{\mathsf{T}}, \delta_1, \ldots, \delta_N]^{\mathsf{T}}$ es un vector de parámetros en el que $\mathbf{w}_{nm}$ son los filtros lineales y δ_n meta-parámetros cuyo valor solo puede ser 1 o -1. De la imagen se toman 6 características, las componentes LUV y los gradientes vertical, horizontal y su magnitud. El objetivo es minimizar la función $\mathcal{L}$ que a su vez es la suma de 3 términos:

$$\text{argmin}_{\omega}(\mathcal{L}_c(\omega) + \mathcal{L}_s(\omega) + \mathcal{L}_t(\omega)) \tag{2.9}$$

- **Clasificación**. Este término sirve para separar los resultado en muestras positivas y negativas asignándoles puntuaciones altas y bajas respectivamente. Esto permite descartar posteriormente las muestras negativas aplicando una umbralización y se expresa mediante la ecuación 2.10.

$$\mathcal{L}_c(\omega) = \gamma_c \left\|\omega\right\|_2^2 + \frac{1}{K} \sum_{i=1}^{K} \max(0,\ 1 - y_i \mathbf{F}(\mathbf{x}_i;\omega))^2 \tag{2.10}$$

  , donde γ_c es un meta-parámetro, $y_c \in \{-1, 1\}$ es una etiqueta positiva o negativa para la muestra $\mathbf{x}_i$, y K es el número de la muestra de entrenamiento.

- **Regularización de forma**. Cada muestra positiva se fuerza a tener una determinada forma definida por la expresión 2.11

$$\mathbf{h}(x, y) = e^{\alpha(1 - \frac{\sqrt{x^2+y^2}}{\beta})} - 1 \tag{2.11}$$

  , donde x, y son las coordenadas de los pixeles con respecto de centro y α, β son meta-parametros que afectan a la suavidad de la pendiente. La regularización debe afectar únicamente a la forma general de la respuesta sin afectar a su escala para no interferir con el termino $\mathcal{L}_c$. El termino de regularización se define mediante la ecuación 2.12.

$$\mathcal{L}_c(\omega) = \frac{\gamma_s}{K_p} \sum_{i|y_i=+1} \sum_{n} \left\| \mathbf{w}_{n\eta_i(n)} * \mathbf{x}_i - (\mathbf{w}_{n\eta_i(n)}^{\mathsf{T}} \mathbf{x}_i)\mathbf{h} \right\|_2^2 \tag{2.12}$$

  , donde K_p es el número de muestras positivas. Sin embargo [3] recomienda que llevar a cabo la optimización de este termino en el dominio de Fourier aprovechando el teorema de la convolución de tal manera que la ecuación 2.12

se transforma en la ecuación 2.13.

$$\mathcal{L}_c(\omega) = \frac{\gamma_s}{K_p} \sum_{i|y_i=+1} \sum_n \mathbf{W}^{\mathsf{T}}_{n\eta_i(n)} \mathbf{S}_i^{\mathsf{T}} \mathbf{S_i} \mathbf{W}_{n\eta_i(n)} \qquad \text{donde} \quad \mathbf{S}_i = (\text{diag}(\mathbf{X}_i) - \mathbf{X}_i\mathbf{H})^{\mathsf{T}} \tag{2.13}$$

Este método se aplica a fin de forzar la forma de la respuesta de las muestras positivas y es una generalización del método propuesto en [6].

- **Regularización temporal**: para reforzar la repetitividad de los puntos detectados a los largo del tiempo se introduce el termino de regularización temporal definido mediante la ecuación 2.14. La finalidad de este termino es la de forzar al detector a devolver respuestas similares en las mismas localizaciones en todas las imágenes de entrenamiento.

$$\mathcal{L}_t(\omega) = \frac{\gamma_t}{K} \sum_{i=1}^{K} \sum_{j \in \mathcal{N}_i} (\mathbf{F}(\mathbf{x}_i;\omega) - \mathbf{F}(\mathbf{x}_j;\omega))^2 \tag{2.14}$$

2.4. Tracking mediante PoI

Uno de los métodos de *tracking* para propósito general más comunes es el *matching* entre PoIs como se menciona en la sección 2.2.1. Esta técnica se utiliza para establecer una correspondencia entre puntos de interés detectados entre cuadros consecutivos de una secuencia (o entre el cuadro actual y puntos en el modelo del objetivo). Gracias a esta correspondencia se puede estimar la posición del objetivo en el cuadro siguiente comparando el cambio de posición de los puntos. En esta sección se van a analizar las características y las técnicas utilizadas por este tipo de *trackers*.

2.4.1. Esquema general

La figura 2.1 muestra el esquema general de los *trackers* basados en la extracción de puntos de interés. Este esquema consta de tres procesos principales: obtención de características, estimación de movimiento y búsqueda [2]. Este esquema general suele tener asociados otros procesos complementarios de cara a optimizar su rendimiento. Además, dependiendo del tipo de aplicación, el diseño se vuelve más complejo en función de las dificultades que el sistema deba para afrontar. Estos procesos complementarios serán discutidos en la sección 2.4.5.

Para inicializar el sistema es necesario proporcionar la posición y tamaño del objetivo en el primer cuadro de la secuencia. Esta información puede ser introducida manualmente mediante la selección de una región de la imagen o puede ser automatizada utilizando detectores que identifican objetivos específicos como vehículos o

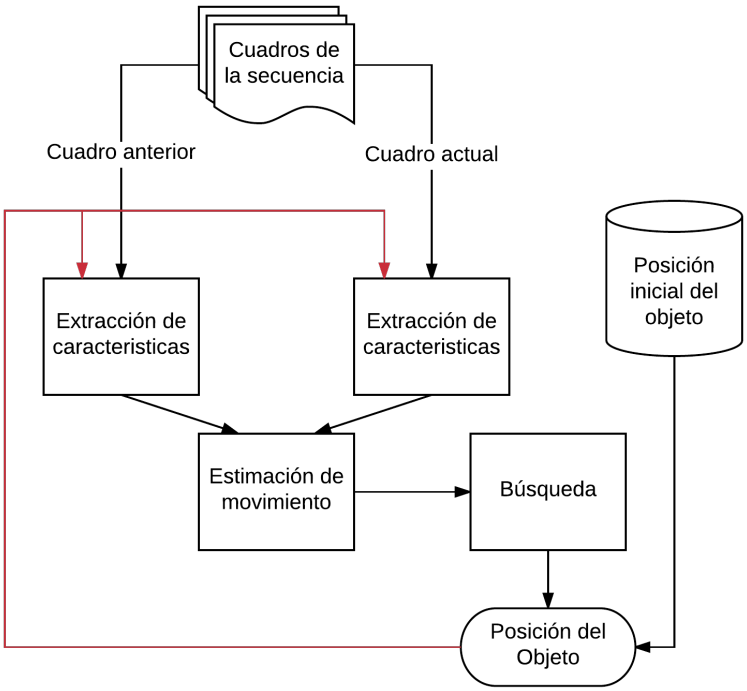

Figura 2.1: Esquema general. Ver detalles en el texto. La flecha de color rojo representa el resultado del algoritmo que a su vez sirve como lazo de retroalimentación.

caras. Por otro lado, como se discutió en la sección 2.2.2, hay diversos métodos para seleccionar este área de inicialización que conllevan la selección en mayor o menor medida de pixeles que no pertenecen al objetivo.

2.4.2. Extracción de características

Este bloque es el encargado de realizar una descripción del objetivo con el fin de hacerlo distinguible del resto de objetos y de su entorno. Estas características son principalmente puntos de interés aunque también pueden complementarse con detectores de *blobs* o de detectores bordes con el fin de contar con mas información que ayude en la descripción del objetivo. Otros métodos complementarios calculan un histograma del área proporcionada para contar con una distribución estadística de los valores de los pixeles del objetivo. Gracia a esta distribución el sistema es capaz de identificar detecciones erróneas.

Dado que este proceso es habitualmente el más costoso computacionalmente, la elección del algoritmo de detección puede llegar a ser crítica si se requiere que el sistema opere en tiempo real . Por último, cabe destacar que la idea principal de

este tipo de *trackers* es establecer una correspondencia entre las características detectadas entre cuadros consecutivos (o entre un cuadro y el modelo). Por este motivo el detector debe ofrecer resultados robustos en cuanto a invariabilidad y repetición. [7] propone como buenas opciones el detector de bordes de Harris y Stephens [8], el detector de esquinas de Shi-Tomasi [9] o el detector de puntos ORB [10].

2.4.3. Estimación de movimiento

Este bloque es el encargado de determinar el movimiento descrito por el objetivo. Una vez obtenido unos putos de interés con las características descritas en la sección 2.4.2, se procede a establecer una correspondencia entre ellos, este proceso es conocido como *mathching*. En función de las correspondencias obtenidas se calcula el movimiento del objeto comparando la variación de la posición de estos puntos de un cuadro a otro. Existen diferentes estrategias para llevar a cabo este cálculo como la obtención de vectores de movimiento o la asignación de puntuaciones a cada correspondencia para determinar su relevancia en el movimiento.

2.4.4. Búsqueda

El proceso de búsqueda tiene como finalidad establecer la localización mas probable del objetivo en el cuadro actual. Esta estimación se realiza mediante métodos de comparación de señales en dos dimensiones como el cálculo de la correlación cruzada o el error cuadrático medio. El procedimiento se basa en comparar un *template*, que puede ser simplemente una sección del cuadro anterior ajustada al objetivo; con un área de búsqueda, que es una sección del cuadro actual en la que se prevé encontrar dicho objetivo.

2.4.5. Procesos complementarios

Dependiendo de la aplicación especifica para la que se vaya a diseñar el *tracker*, se deben tener en cuenta una serie de dificultades descritas en las sección 2.2.3. Algunos de estos procesos complementarios son:

- **Recuperación**. La motivación de este bloque es amplia. En primer lugar actúa como detector de fallos en el seguimiento y ofrece un método de recuperación para aumentar la fiabilidad del algoritmo. Por otro lado resulta efectivo en situaciones adversas como movimientos rápidos u oclusiones. Finalmente, resulta imprescindible en *trackers* diseñados para analizar secuencias de larga de duración en la que el objeto se puede perder numerosas veces o incluso llegar a salir de la escena.

- **Corrección de perspectivas**. Este bloque estaría destinado a ofrecer protección ante cambios de perspectiva, que es una característica habitual en escenarios realistas. El método aprovecha las correspondencias entre los puntos de interés detectados para realizar una homografía entre los dos cuadros. La homografía transforma el objeto del cuadro anterior otorgándole la perspectiva que tiene en el cuadro actual, maximizando de esta manera las probabilidades de ser encontrado correctamente en el proceso de búsqueda.

2.5. Conclusiones

Como se ha visto en este capitulo el diseño apropiado de un *tracker* lleva asociado un gran numero de variables. Un punto clave es determinar el propósito del algoritmo ya que cuantas más dificultades se prevea afrontar mas complejo resulta su el diseño e implementación. Por otro lado hay que prestar atención a los métodos que se van a utilizar para llevar a cabo los diferentes procesos teniendo en cuenta requisitos impuestos como el tiempo de ejecución o la capacidad de computación disponible.

Los algoritmos de *tracking* basados en la detección de PoIs ofrecen una solución competente debido a su gran versatilidad, flexibilidad y buenos resultados. La detección de puntos de interés tiene diversas aplicaciones pero en *tracking* se busca obtener puntos con una alta estabilidad y repetitividad. En este sentido TILDE [3] ofrece buenos resultados siendo el algoritmo elegido par detección en este trabajo fin de grado.

Capítulo 3

Diseño y desarrollo

3.1. Introducción

En esta sección se describe el algoritmo desarrollado explicando cada etapa y su motivación. Dentro de la clasificación de los tipos de *trackers* realizada en el Estado del Arte 2, se podría considerar que el algoritmo propuesto encaja con la descripción de *trackers* mediante correspondencia de PoIs. Efectivamente el método saca provecho de esta operación pero difiere en el papel que juegan estas correspondencias en el proceso de seguimiento.

El tracker propuesto utiliza las correspondencias entre los PoI para determinar un área de interés más precisa. De esta forma el proceso de búsqueda se realiza sobre un área muy concreta. Esto además favorece ademas a la velocidad de ejecución debido al menor número de comparaciones.

Para terminar la sección, se van a discutir otros procesos complementarios al algoritmo que pueden representar un interesante trabajo futuro.

3.2. Diseño del algoritmo

El algoritmo propuesto se basa en la detección de PoI en cada par de cuadros consecutivos de la secuencia. Una vez obtenidos estos puntos se establece una correspondencia entre ellos a fin de determinar la nueva posición del objetivo. Este paso se denomina estimación de movimiento como se explicó en la sección 2.4.3.

Gracias a esta información se aumenta la eficiencia y precisión del algoritmo centrando la búsqueda del objetivo en una zona especifica del cuadro siguiente. Repitiendo este proceso de manera iterativa se busca seguir al objetivo a lo largo de la secuencia identificando su posición en cada cuadro.

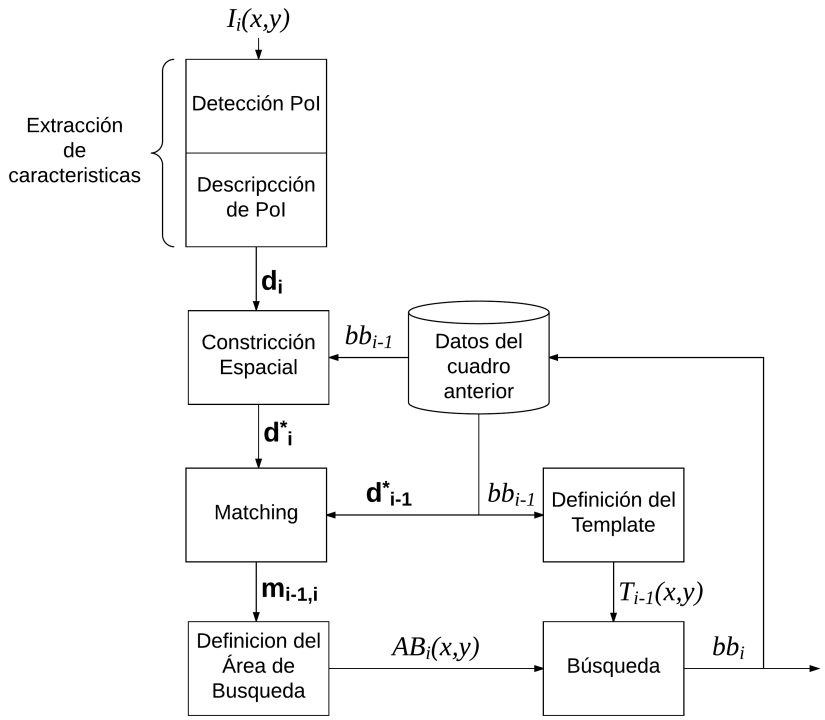

Figura 3.1: Diagrama de bloques del algoritmo diseñado. Ver texto para una descripción pormenorizada de los módulos involucrados.

El esquema general del algoritmo propuesto se incluye en la figura 3.1, donde se sigue la siguiente nomenclatura:

- I_i: Cuadros que forman la secuencia.
- $bb_i = \{x_i^1, y_i^1; x_i^2, y_i^2\}$: coordenadas que definen el *bounding box*.
- $\mathbf{d_i}$: Descriptores de los puntos de interés
- $\mathbf{d_i^*}$: Selección de los descriptores en función de las coordenadas de bb_i.
- $\mathbf{m_{i-1,i}}$: Correspondencia entre los puntos.
- AB_i: Área de búsqueda, zona de la cuadro actual I_i en la que se busca el *template*.
- T_{i-1}: *Template*, zona de la cuadro anterior I_{i-1} que se busca en el área de búsqueda.

3.2.1. Datos de entrada

Para iniciar el algoritmo se debe proporcionar el primer par de cuadros de la secuencia $I_1(x, y), I_2(x, y)$ así como unas coordenadas iniciales bb_1 para señalar la posición del objetivo en el primer cuadro de la secuencia. Durante la primera iteración del algoritmo, los dos cuadros se procesan de manera paralela calculando PoIs y descriptores para cada uno de ellos para después realizar la constricción espacial y la correspondencia. El siguiente paso es definir el área de búsqueda en función de estas correspondencias. Una vez extraído el *template* y el área de búsqueda se procede a determinar la posición precisa del objetivo en el segundo cuadro.

Como resultado de esta primera iteración se obtienen las coordenadas que indican la posición del objetivo en el segundo cuadro, es decir, bb_2. En las sucesivas iteraciones el único dato de entrada será el siguiente cuadro de la secuencia tal como se muestra en la figura 3.1. En la base de datos se almacena la posición obtenida y los puntos detectados en cada iteración con el fin de utilizarlos en la siguiente.

3.2.2. Extracción de características y constricción espacial

Para obtener las características del par imágenes se utiliza el algoritmo TILDE explicado en la sección 2.3.5. TILDE [3] proporciona un primer análisis de las imágenes extrayendo de ellas las coordenadas de los PoI. Este método ofrece un detector robusto a cambios meteorológicos y de iluminación por lo que el algoritmo de *tracking* resultante hereda potencialmente estas ventajas. Para realizar una correspondencia se deben obtener previamente los descriptores $\mathbf{d_i}$. Este proceso se lleva a cabo usando el descriptor SIFT explicado en la sección 2.3.4. Los descriptores sirven para identificar cada uno de los PoI de manera unívoca de cara a optimizar la correspondencia.

El siguiente paso es hacer una selección apropiada de este conjunto de descriptores en función de las coordenadas definidas por bb_{i-1}. Esta selección se realiza tomando todos los puntos que se encuentran dentro del área de interés. El área de interés es un rectángulo cuyo centro coincide con el de el *bounding box* y cuyas dimensiones han sido incrementadas. Tras este proceso se obtiene un subconjunto de descriptores $\mathbf{d_i^*}$ representantes de los PoI en torno al objetivo, obviando todos los demás puntos de la imagen que pudieran interferir en la identificación.

3.2.3. *Matching*

El objetivo del *matching* es establecer un correspondencia entre dos cuadros consecutivos de la secuencia con el fin de obtener información sobre su desarrollo espacial. Por otra parte, gracias a la constricción espacial realizada en el paso anterior se consi-

gue centrar esta información en el objetivo descartando otras zonas de la imagen. Una vez realizada la correspondencia se obtiene el vector $\mathbf{m_{i-1,i}}$ que indica que puntos de I_{i-1} e I_i se corresponden. Cada correspondencia indica la aparición de una misma característica (borde, extremo, esquina, etc.) en ambos cuadros. Dado que ademas conocemos las posición de estos puntos podemos calcular la dirección y la magnitud de su movimiento.

El verdadero reto en este paso del algoritmo es conseguir hacer una diferenciación eficaz entre los puntos que pertenecen al objetivo y los que no

3.2.4. Definición del área de búsqueda

Una vez obtenidas las correspondencias entre puntos se puede estimar con mayor precisión la zona hacia la que se ha movido el objetivo. Esta estimación se realiza de diversas formas en función de los resultados obtenidos en el paso anterior. La idea general es tomar el área delimitada por la nube de puntos con correspondencias positivas. Este área se traza mediante un rectángulo por lo que es condición indispensable la existencia de al menos dos correspondencias positivas que definan sus esquinas. Se diferencian los siguientes casos:

1. **Hay menos de dos correspondencias**: Este es el caso de menor eficiencia del algoritmo ya que es incapaz de determinar un buen área de búsqueda. En consecuencia se designa como área de búsqueda el cuadro actual completo.

2. **El área de búsqueda obtenido no contiene el *bounding box***: El *bounding box* del cuadro anterior bb_{i-1} es la zona que indica la ultima posición conocida del objetivo. Al buscar dicho objeto en el cuadro actual, resulta lógico hacerlo en una zona alrededor de su última posición. Sin embargo, sería un error de diseño no tener en cuenta que el objetivo puede no haberse movido en absoluto manteniendo la misma posición que en el cuadro anterior. Por este motivo un área de búsqueda coherente debe contener el ultimo *bounding box* obtenido. En el caso de no cumplirse esta condición se toma como área de búsqueda el área de interés que no es mas que una versión expandida uniformemente del *bounding box*.

3. **El área de búsqueda obtenido contiene el *bounding box:*** Este caso representa el de mayor rendimiento del algoritmo ya que el área de interés obtenido es menor y mas preciso reduciendo así el número de comparaciones necesarias para determinar la nueva posición del objetivo,

3.2.5. Definición del *template*

En este paso se determina la imagen que se desea encontrar en el área de búsqueda. El *template* es extraído del cuadro anterior. Esta extracción se realiza de acuerdo al *bounding box* que es el que indica la posición y dimensiones del objetivo en cada cuadro. Es importante aclarar la diferencia entre *bounding box* y *template*. El primero hace referencia a dos coordenadas del cuadro gracias a las cuales se puede localizar el objetivo en dicho cuadro. Por otro lado, el *template*, es la imagen de cuadro anterior que se extrae tomando el área que delimita por el rectángulo definido por estas coordenadas.

3.2.6. Búsqueda

El proceso de búsqueda requiere dos datos de entrada, el *template* o elemento buscado; y el área de búsqueda. El algoritmo está diseñado de tal forma que el área de búsqueda sea lo mas pequeño posible agilizando de esta manera la ejecución. El tamaño y por tanto la precisión del área de búsqueda extraída dependerá de las correspondencias obtenidas en la etapa de *matching*.

Para la búsqueda se calcula la correlación cruzada y la suma de las diferencias el cuadrado que se definen:

- **Correlación cruzada**. sean $f[n]$ y $g[n]$ dos funciones discretas, la correlación cruzada ($\star$) se define según la ecuación 3.1

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^*[m]g[m+n] \tag{3.1}$$

  donde f^* es el conjugado complejo de f . Esta operación se puede calcular de una manera más eficiente haciendo uso del Teorema de la correlación cruzada indicado en la ecuación 3.2

$$\mathcal{F}\{(f \star g)[n]\} = \mathcal{F}\{f[n]\}^*\mathcal{F}\{g[n]\} = F^*[u]\,G[u] \tag{3.2}$$

  , donde $\mathcal{F}$ representa la transformada de Fourier. Calculando la transformada inversa de Fourier ($\mathcal{F}^{-1}$) en ambos lados de la igualdad la ecuación 3.2 se transforma en la ecuación 3.3.

$$(f \star g)[n] = \mathcal{F}^{-1}\{F^*[u]\,G[u]\} \tag{3.3}$$

- **Suma de las diferencias al cuadrado**. Este método consiste en calcular la

magnitud de la distancia entre las dos señales en cada punto. para ello se utiliza la formula descrita en la ecuación 3.4

$$h[n] = \sum_{m=-\infty}^{\infty} (f[n] - g[m+n])^2 \tag{3.4}$$

Este método al igual que el anterior hace un barrido de una señal sobre la otra recorriendo todas las posiciones posibles. El resultado es otra señal $h[n]$ en cuyo valor mínimo se localiza la posición de máxima similitud entre las dos señales.

Un área de búsqueda óptimo es aquel ligeramente mayor que el *template* lo que indicaría que el objetivo ya ha sido localizado y por tanto el bloque de búsqueda solo se encargaría de fijar su posición precisa. En el otro extremo, el peor caso es aquel en que el área de búsqueda es el cuadro actual completo. En este caso no se ha podido determinar una posición especifica del objeto mediante el uso de puntos de interés por lo que el bloque de búsqueda pasa a tratar de recuperarlo sin información adicional (es decir, en este caso se usa el algoritmo NCC [11]).

3.3. Implementación del algoritmo

En esta sección se va a explicar como se ha implementado el algoritmo así como otros detalles técnicos. El sistema se ha programado íntegramente en el lenguaje y entorno de programación MATLAB[1]. La interfaz del algoritmo está implementada en el archivo **main.m** el cual se encarga de invocar una serie de funciones que realizan las tareas básicas. Algunas de estas de estas funciones son la implementación en MATLAB de algoritmos externos usados por el *tracker* como SIFT [5] o TILDE [3].

3.3.1. Instalación

Para el funcionamiento del programa se requiere la instalación previa del algoritmo TILDE en el entorno MATLAB. Su implementación puede ser descargada desde el sitio web del proyecto TILDE [2]. Una vez descargado y descomprimido el fichero, se deben crear los directorios siguiendo la estructura mostrada en la Figura B.1. Los *scripts* del *tracker* deben estar alojados en el directorio **tracker** mientras que el directorio **sequences** contendra subdirectorios con las secuencias que se desea procesar. Estas secuencias deben estar divididas en archivos de imagen correspondientes a los cuadros. Cabe destacar que la implementación del algoritmo SIFT para MATLAB

[1] www.mathworks.com
[2] cvlab.epfl.ch/research/tilde

ya esta incluida en el paquete TILDE por lo que no son necesarias instalaciones adicionales.

3.3.2. Opciones e inicialización

El programa principal **main.c** comienza preparando rutas, variables y cargando los archivos necesarios para los posteriores procesos. Las opciones son parámetros ajustables por el usuario para adaptar el algoritmo a determinados escenarios mejorando así su rendimiento en función de la situación. Estas opciones y su papel en la ejecución serán explicadas más adelante.

En la inicialización se ponen en marcha dos bucles, uno que recorre todas las secuencias alojadas en el directorio **sequences** y otro para recorrer cada cuadro de dichas secuencias. Para analizar una secuencias concreta o fragmentos de secuencias, basta con cambiar los indices de estos bucles. Además, la variable `f_start` permite seleccionar el cuadro con el que comenzar la secuencia. Cada vez que empieza a procesar una secuencia se produce una nueva inicialización.

Esta inicialización se encarga de calcular el número y tamaño de los cuadros, y de fijar la posición inicial del objetivo. Para fijar esta posición de puede activar la variable `bb_manual`, lo que permite seleccionar el *bounding box* manualmente con el ratón. En caso de desactivar esta opción, el programa lee esta posición inicial del archivo **groundtruth** de la secuencia en caso de que exista.

Tras esta lectura se fijan las dimensiones y coordenadas del *bounding box* y del área de interés. Como se explicó en la sección 3.2.2, este área de interés es una versión expandida del *bounding box*. La variable `inc_AoI` determina la magnitud de esta expansión.

3.3.3. Detección de puntos de interés

El siguiente proceso es la extracción de características del cuadro. Para esto, el programa distingue entre la primera y sucesivas iteraciones. En la primera iteración se extraen las características de los dos primeras cuadros mientras que en el resto de la ejecución solo se analiza el cuadro actual. La detección de puntos de interés se realiza llamando a la función `cargaPTILDE`. Debido al prolongado tiempo de ejecución del algoritmo TILDE, esta función está diseñada para guardar los puntos detectados en un archivo **.mat**. En cada llamada a la función se comprueba la existencia de este archivo y en caso de existir, los puntos no se calculan sino que se cargan agilizando de esta forma la ejecución del programa.

Tras obtener los puntos TILDE se hace un selección de ellos mediante la función

`selectTILDEPoints` siguiendo tres criterios:

- **Constricción espacial**. Se descartan todos los puntos que estén fuera del área de interés. Este criterio esta directamente relacionado con la variable de inicialización `inc_AoI`.
- **Umbral de puntuación**. Junto con las coordenadas de los puntos detectados, `cargaPTILDE` devuelve una puntuación para estos en orden descendente. Como se menciono en la sección 2.3.5, para obtener los mejores puntos basta con descartar los puntos por debajo de cierto umbral. Este umbral se fija mediante la variable `umbral_TILDE`.
- **Número de puntos**: En las ocasiones en las que es los dos criterios anteriores no son lo suficientemente restrictivos, se puede determinar el numero máximo de puntos que se desea obtener. Esto es especialmente útil en situaciones con entornos muy ricos en los que se detectan muchas características alrededor del objetivo, el resultado son los n puntos con mejor puntuación. Este parámetro se fija mediante el valor de la variable `num_PTILDE`.

Finalmente en este bloque del programa se realiza la descripción de los puntos de interés utilizando la función `describePoints`. Esta función hace uso de la implementación del descriptor SIFT, pasando los puntos a un formato adecuado para su procesamiento.

3.3.4. Correspondencia, definición de áreas y búsqueda

Esta sección es la parte central del programa. Tras obtener los datos necesarios se pasa a establecer una relación entre los cuadros para determinar la nueva posición del objetivo. El primer paso el realizar la correspondencia entre los PoI obtenidos en el paso anterior. Para este fin se ha implementado la función `myMatch`. Esta función se encarga de llamar a la función de matching de SIFT y consta de dos modos de funcionamiento determinados por la variable `check`.

- **Checkback desactivado**. En este modo la función actúa como un mero puente entre el **main.m** y el *matching* de SIFT. El programa principal recibe un vector que indica que correspondencia hay entre `f1` y `f2` haciendo referencia a sus indices. SIFT ademas devuelve la distancia entre los puntos al cuadrado. La función ordena los puntos en función de estas distancias en orden ascendente, haciendo que los primeros valores del vector `my_matches` sean las mejores correspondencias.

- **Checkback activado**. Con este modo activado la función hace una doble llamada selectiva al *matching* de SIFT. El *matching* habitual funciona estableciendo la correspondencia de todos los puntos `f1` con todos los puntos `f2`. En este modo se toma uno a uno cada punto de `f1` haciendo la correspondencia solo con los puntos de `f2` mas cercanos a el. Esta cercanía se define mediante un circulo cuyo radio corresponde al indicado en la variable `radio`. tras esta primera correspondencia este proceso se vuelve a repetir en el sentido opuesto a modo de comprobación. Es decir, se toman uno a uno los puntos de `f2` con correspondencia positiva, se eligen los puntos `f1` cercanos y se realiza el *matching* con SIFT. Si esta segunda correspondencia no coincide con la primera, se descarta.

Una vez obtenidas las correspondencias se determina el *template* y el área de búsqueda según lo explicado en las secciones 3.2.4 y 3.2.5 mediante la función `getTemplates`. El paso final es buscar el *template* en el área de búsqueda. Este proceso sido ha detallado también con anterioridad en la sección 3.2.6 aunque cabe destacar que la implementación del algoritmo de comparación del que hace uso la función `getBoundingBox` ha sido tomada de un desarrollador externo[3].

3.4. Desarrollos no incluidos

En esta sección se incluyen algunos procesos complementarios al algoritmo propuesto. Estos procesos has sido desarrollados y probados en versiones anteriores del programa aunque finalmente no han sido incluidos en la versión presentada en este trabajo fin de grado. El motivo en la mayoría de los casos ha sido procurar mantener la sencillez del algoritmo ya que estos procesos requieren una implementación compleja.

3.4.1. Historial de PoI detectados

En el algoritmo propuesto se calculan los puntos TILDE del cuadro actual para posteriormente realizar la correspondencia con los puntos del cuadro anterior. Este proceso se realiza para cada nuevo cuadro descartando automáticamente PoIs con correspondencias negativas, sin importar su buena localización en el cuadro o las correspondencias positivas obtenidas con anterioridad. Para mejoraras los puntos candidatos y así las correspondencias resultantes se desarrolló un sistema que guarda un historial de PoIs detectados en cuadros anteriores.

[3] es.mathworks.com/matlabcentral/fileexchange/24925-fast-robust-template-matching

Para ello se introduce el concepto de tiempo de vida, el cual es un valor asociado a cada punto de interés que determina el número de cuadros sin correspondencia positiva que deben transcurrir para ser definitivamente descartado. Tras realizar la correspondencia de los nuevos PoIs se asigna un tiempo de vida bajo a los puntos con correspondencia negativa mientras que a las correspondencias positivas se les asigna un valor mayor. En la siguiente iteración del algoritmo la correspondencia se realiza entre todos los puntos aun vivos y los nuevos detectados en el cuadro actual. Una nueva correspondencia positiva aumenta el tiempo de vida en uno mientras que en caso contrario se reduce. Los puntos que lleguen a tiempo de vida cero son finalmente descartados.

La idea principal de este proceso es preservar puntos cuya correspondencia ha sido positiva a lo largo de varias secuencias evitando que sean descartados por correspondencias negativas asiladas.

3.4.2. Asignación de confianzas

El algoritmo propuesto actualmente utiliza las correspondencias entre PoIs para determinar el área de búsqueda del objeto. Este proceso es complementario al anterior y se basa en la utilización de las correspondencias positivas calcular la nueva posición del objeto.

- **Mapa de confianzas**: Este método se basa en asignación de una puntuación o confianza a cada correspondencia de puntos en función de criterios como su posición dentro del *bounding box* o el número de correspondencias previas.
- **Vector de movimiento**: Para este método es necesario trazar vectores entre cada par de puntos con correspondencia positiva. El origen, magnitud y orientación de cada vector se calcula mediante las coordenadas de los puntos con correspondencia positiva tomando como origen las de los puntos del cuadro anterior. Finalmente el movimiento se estima promediando estos vectores o ponderandolos por unas puntuaciones similares a las del método anterior

3.4.3. Homografías

Una homografía es una transformación en la perspectiva de la imagen. Aprovechando la correspondencia de puntos que calcula el algoritmo se construye una homografía entre cada par de cuadros de la secuencia con el objetivo de establecer un template adaptado que favorezca el proceso de búsqueda.

La secuencias de video suele verse afectadas por cambios de perspectiva debido al movimiento de la cámara o a la rotación de los objetos. Esta característica representa

una dificultad para el proceso de búsqueda ya que al realizar un cambio de perspectiva el objeto puede sufrir cambios en la sus dimensiones relativas.

El método de corrección de perspectiva mediante homografías propone hacer una transformación en las imágenes de tal manera que el cuadro anterior adquiera la perspectiva del cuadro actual. Dado que del cuadro anterior se extrae el *template,* este proceso contribuye a que el proceso de búsqueda se mas preciso en estos casos.

Capítulo 4

Evaluación

4.1. Introducción

Una vez implementado el algoritmo es necesario comprobar su funcionamiento en todo tipo de situaciones con el fin de determinar su eficiencia. Este proceso es necesario para detectar sus puntos fuertes y débiles a fin de solventarlos en un trabajo futuro. En este capítulo es expondrán los experimentos realizados para determinar la eficacia del *tracker*. Finalmente se hará un análisis de los resultados.

4.2. Marco de evaluación

4.2.1. *Dataset*

El *Visual Object Tracking* (VOT) es una comunidad de investigadores dedicada a la evaluación de algoritmos de seguimiento de objetos. El VOT convoca anualmente un concurso con el fin de establecer una plataforma de discusión para el intercambio de métodos y resultados. Esta convocatoria conocida como VOT *Challenge* pone a disposición un *dataset* de secuencias que recogen un amplio abanico de situaciones desafiantes para un *tracker*.

Se ha elegido este *dataset* para realizar las pruebas ya que resulta accesible públicamente ademas de ser ampliamente conocido en el área del *tracking*. Junto con cada secuencia se proporciona un archivo **groundtruth** en el que se recogen los resultados esperados del *tracking* a modo de solución. Gracias a esto se puede cuantificar la eficiencia del algoritmo mediante simple comparación del resultado con el **groundtruth**. Estos detalles serán discutidos en profundidad en la sección 4.2.2.

4.2.2. Métricas

En esta sección se van a discutir los métodos utilizados para cuantificar los resultados obtenidos. Como se comentó en la sección 4.2.1, el *dataset* del VOT Challenge proporciona los resultados esperados del proceso de seguimiento para cada secuencia. De esta manera, para estimar la precisión del algoritmo se ha procedido a calcular el solapamiento del *bounding box* proporcionado con el detectado por el *tracker* en cada cuadro.

Para esto se ha hecho uso de una de las métricas mas habituales en este campo, la *Sequence Frame Detector Accuracy* (SFDA), definida en [12] y cuya expresión se define mediante la ecuación 4.1.

$$\phi_i = \frac{G_i \cap D_i}{G_i \cup D_i} \tag{4.1}$$

, donde G_i y D_i denotan los *boundig box* del **groundtruth** y el detectado respectivamente, y ϕ_i denota la tasa de solapamiento para el cuadro i. Esta expresión proporciona una medida de precisión por cuadro, por lo que si se quiere una valoración global de la secuencia basta con calcular la media de ϕ_i como indica la ecuación 4.2.

$$\Theta = \frac{1}{N} \sum_{i=1}^{N} \phi_i \tag{4.2}$$

Sin embargo, una de las dificultades a la hora de comparar resultados ha sido el formato del **groundtruth** que define la forma del *bounding box*. El **groundtruth** de cada secuencia proporciona cuatro coordenadas para cada cuadro para determinar el bounding box. Esto significa que *el bounding box* puede aparecer rotado en cualquier ángulo y sentido con el fin de adaptarse mejor a la forma y posición del objetivo. Sin embargo, el *tracker* propuesto define el *bounding box* mediante solo dos coordenadas por lo que no contempla la posibilidad de rotaciones.

4.3. Pruebas y resultados

Para determinar la eficiencia del algoritmo se ha procedido a comparar el área de solape del *bounding box* detectado con el proporcionado por **groundtruth**. Para esta comparación se ha calculado la tasa de solapamiento para cada cuadro de la secuencia como se ha explicado en la sección 4.2.2. De esta forma cuanto más se adapte el tamaño y la posición del *bounding box* detectado mayor será la tasa de solapamiento, llegando a tomar el valor 1 en caso de ser idénticos y 0 si no se solapan en ningún punto.

4.3.1. Evaluación

Ademas de la tasa de solapamiento se han tenido en cuenta otros factores como el tiempo de ejecución por cuadro y el numero de fallos por secuencia. La tasa de solapamiento descrita en la ecuación 4.1, también es útil para contabilizar el número de fallos. Se cuenta como fallo aquel cuadro en el que la tasa de solapamiento es 0 indicando esto que el *bounding box* no señala, ni siquiera parcialmente, la posición del objetivo.

De cara a analizar el *tracker* de la manera más imparcial posible se le proporciona la posición inicial del objetivo a través del **groundtruth** como parámetro de entrada. Esto garantiza un punto de partida idéntico en todos los experimentos ya que inicializaciones distintas producen resultados potencialmente distintos.

Tras esta inicialización se calcula cuadro a cuadro la tasa de solapamiento y el tiempo de ejecución hasta el final de la secuencia o hasta que se produce un fallo, es decir hasta que se pierde el objetivo. En este caso el algoritmo de evaluación incrementa en 1 el contador de fallos e inicia el proceso de recuperación. Este proceso procede igual que la inicialización, es decir, lee la posición correcta del **groundtruth** y descarta la detectada por el *tracker.* De esta forma resulta posible realizar un análisis completo aunque el algoritmo pierda el objeto en ciertas ocasiones.

Esta evaluación mide el rendimiento del *tracker* mediante dos indicadores, la tasa de solapamiento global (ecuación 4.2) y el numero de fallos totales. Para el calculo de la de la tasa global de solapamiento se han ignorado los cuadros en los que se han producido fallos con el fin de evitar perjudicar el valor de esta tasa, ya que estos fallos se contabilizan aparte.

Por ultimo cabe destacar que el formato del *bounding box* contemplado en el **groundtruth** esta constituido por más puntos de los que maneja el *tracker.* Por este motivo ha sido necesario adaptar el *bounding box* leído del **groundtruth** (al inicializar o al recuperarse de un fallo) al formato utilizado por el *tracker.* Para ello se han rectificado las posibles rotaciones del *bouding box* pero manteniendo su centro y sus dimensiones.

4.3.2. Comparación con otros algoritmos

Para evaluar el rendimiento del algoritmo propuesto se ha realizado una comparación con otros *tracker*s. El VOT publica anualmente los resultado del concurso facilitando el acceso a los datos de evaluación de los algoritmo presentados. Además se proporciona un artículo [13] que utiliza estos datos para ordenar los algoritmos en función de su eficacia.

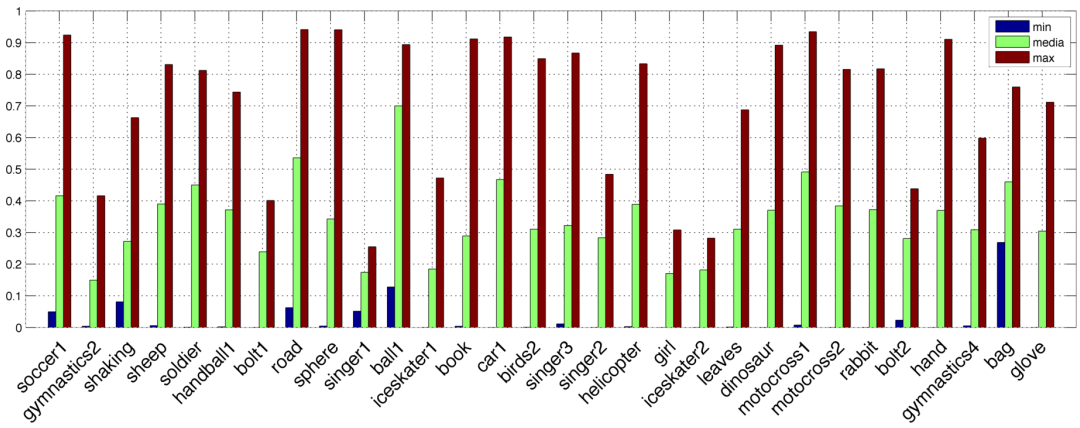

Figura 4.1: Precisión del *tracker* propuesto. La gráfica recoge el valor medio de la precisión para toda la secuencia así como sus valores máximo y mínimo. *dataset VOT2016.*

A fin de poder ubicar el algoritmo propuesto según este criterio, se han seleccionado dos métodos de entre los 62 presentados en el VOT2015. Concretamente se han seleccionado MDnet y NCC que son los algoritmo que obtuvieron el primer y último puesto respectivamente.

Para su evaluación se ha procedido de manera idéntica a la aplicada al *tracker* propuesto, detallada en la sección 4.3.1.

4.3.3. Resultados

Los resultados sobre la precisión del algoritmo se exponen en la figura 4.1. Esta gráfica representa el solapamiento global entre el bounding box detectado y la solución del **groundtruth**. Como se puede observar el solapamiento medio oscila entre el 40 y el 50 % lo que indica que los *bounding box* detectados se solapan al menos en la mitad de su área con la detección óptima.

El **groundtruth** maneja para el *bounding box* áreas rectangulares con cualquier tipo de rotación. Además adapta su tamaño en cada cuadro para ajustarlo al tamaño del objetivo, es decir, es robusto a cambios de escala. Por otro lado el algoritmo propuesto padece la limitación del *bounding box* sin rotaciones además de no estar preparado para detectar cambios de escala en la secuencia. Esto tiene como consecuencia una peor adaptación al objetivo con la consiguiente perdida de precisión.

La figura 4.2 muestra la comparación del solapamiento global de los tres algoritmo. En esta gráfica se observa, como era de esperar, que MDnet (el ganador del VOT *Challenge* 2015) ofrece los mejores resultados en la mayoría de las secuencias. Por otro lado NCC (último puesto del concurso) ofrece en general los resultados más

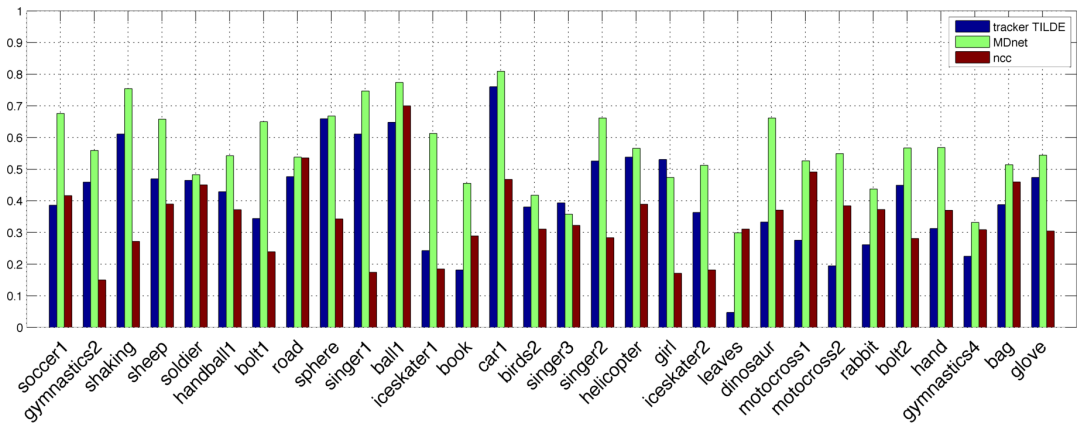

Figura 4.2: Comparativa de la precisión entre el *tracker* propuesto (*tracker* TILDE) y los *tracker* escogidos.

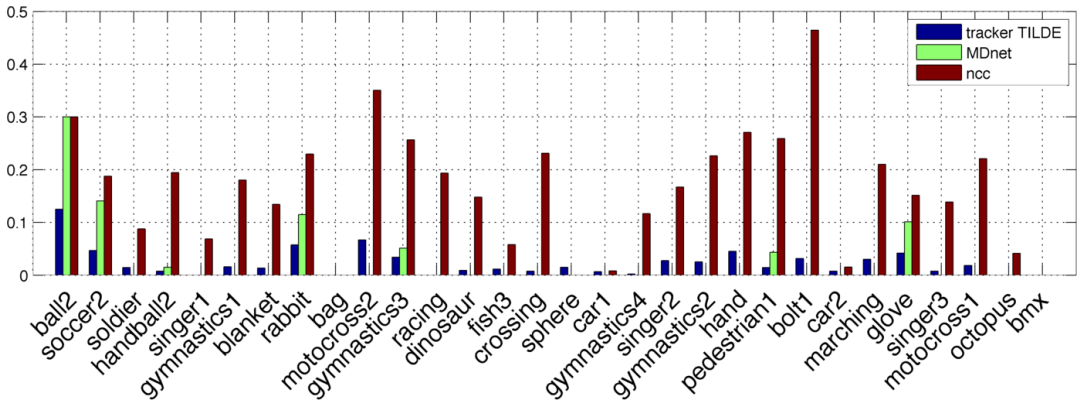

Figura 4.3: Comparativa del porcentaje de fallos por secuencias.

bajos. Por último el algoritmo propuesto presenta un rendimiento intermedio.

En la figura 4.3 se representa el porcentaje de fallos por secuencia. Se puede apreciar la enorme diferencia entre el algoritmo MDnet y NCC, estando el tracker propuesto mas cercano a los resultado de MDnet e incluso superándolo en algunas secuencias. Esto indica, que aunque la precisión de la detección es moderada, el algoritmo es capaz de estimar la posición aproximada en casi todas las situaciones sin sufrir apenas pérdidas del objetivo

Por último, la figura4.4 muestra los tiempos medios de procesamiento por cuadro de cada secuencia. Hay que señalar que diversos factores intervienen en este tiempo de procesamiento aunque el más determinante es el tamaño del cuadro. Cuadros con una gran resolución son evidentemente más lentos de tratar. Si además la secuencia tiene un duración prolongada el tiempo de ejecución del algoritmo se multiplica.

En este caso los algoritmos comparados muestran una relación directa entre su

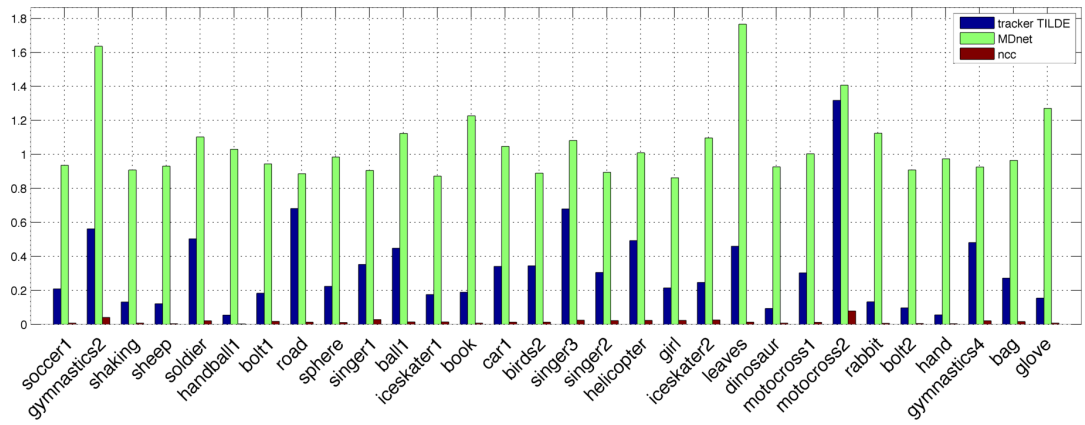

Figura 4.4: Comparativa del tiempo de procesamiento.

eficacia y su tiempo de ejecución. En este terreno el algoritmo NCC es el gran vencedor arrojando resultados de fracciones de segundo para la todas las secuencias. Esto hace ser un candidato a considerar en aplicaciones en tiempo real en las que no se necesite un detección tan precisa. Por otro lado el *tracker* TILDE y MDnet manejan tiempo de procesamiento mucho más prolongados como consecuencia de su mayor complejidad.

4.4. Conclusión

Tras haber realizado la evaluación del algoritmo propuesto se pueden extraer un serie de conclusiones que se detallan a continuación. Pese a su sencillez el *tracker* propuesto presenta un rendimiento competente. Esta conclusión se extrae al realizar la comparación de con algunos de los algoritmos propuestos en el VOT *Challenge.* El *tracker* propuesto supera la eficacia del algoritmo NCC y se acerca bastante al MDnet en términos de detecciones positivas.

Por otro lado su precisión resulta moderada. Este es debido sobre todo a la poca flexibilidad que tiene a la hora de definir el *bounding box*. Este punto representa un interesante trabajo futuro.

Cabe destacar también que el algoritmo presenta un rendimiento similar en todas las secuencias del *dataset*. Estas secuencias están especialmente diseñadas para presentar dificultades como las mencionadas en el capitulo 2.2.3. Este resultado satisfactorio sobre el *dataset* indica que el algoritmo representa una solución robusta al problema.

Por último mencionar los resultados del tiempo de procesamiento. Esta característica no es uno de los puntos fuertes de los algoritmos, sin embargo arroja resultado intermedios entre los dos métodos comparados. Esto hace considerar el *tracker* TIL-

Figura 4.5: Seguimiento del objetivo en la secuencia **dinosaur**. En rojo el *bounding box* deseado, en verde la detección realizada. Cuando ambos rectángulos dejan de solaparse se considera un fallo en la detección.

DE como una solución de compromiso entre el rendimiento y la velocidad de procesamiento. En cualquier caso estos tiempo de procesamiento imposibilitan al algoritmo para trabajar con secuencias en tiempo real.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

El objetivo principal de este trabajo era el diseño y la implementación de un *tracker* de propósito general que aprovechara las características que ofrece el detector de PoI TILDE. Para ello se definieron en el capitulo 1 una serie de objetivos que se van a comentar a continuación

1. **Estudio detallado del estado del arte**. Gracias a esta fase se ha conseguido adquirir una visión global de los tipos de algoritmos desarrollados así como de sus características. Además de ser un excelente punto de partida, este primer objetivo ha servido para decidir que estrategias de diseño adoptar, probando alternativas menos exploradas como la utilización de los PoIs para determinar el área de búsqueda.

2. **Desarrollo de un algoritmo de seguimiento.** Este objetivo representa la principal motivación del trabajo. En esta fase se diseñaron diferentes algoritmos optando por una solución de compromiso entre el rendimiento y la sencillez. Sin embargo estas otras soluciones y procesos complementarios deben ser considerados de cara a mejorar el rendimiento en un trabajo futuro.

3. **Evaluación de la solución propuesta**. La evaluación y comparación de los resultados ha permitido determinar la viabilidad y eficacia del algoritmo. Este objetivo es vital para determinar los pasos siguientes de cara a mejorar determinados aspectos del algoritmo.

4. **Generación de conclusiones**. Las conclusiones extraídas sirven como balance global del trabajo realizado. Así mismo, determinan la consecución de los objetivos propuestos.

5.2. Trabajo futuro

Como se ha observado en el apartado de evaluación (capitulo 4), el algoritmo propuesto presenta una solución competente. Sin embargo cabe tener en cuenta sus limitaciones, de cara a solventarlas; y sus puntos fuertes, considerándolos aciertos en el diseño. De esta forma se definen las siguientes lineas de trabajo futuro

- **Adaptación del *bounding box***. Como se observó en la evaluación del algoritmo, la solución propuesta ofrece buenos resultado pero con una precisión moderada. Este hecho es una consecuencia de las limitaciones en el formato del *bounding box*. En la implementación actual, el *bounding box está* definido de la manera más sencilla posible. Sin embargo, de cara a solventar esta precisión moderada se puede estudiar la implementación de un *bounding box* más preciso así como de un sistema para detectar cambios de escala en el objetivo.

- **Recuperación automática**. Otra de las limitaciones importantes del *tracker* es su carencia de mecanismos para la recuperación del objetivo. Esto hace que en esta fase del desarrollo sea poco viable para su implementación en aplicaciones reales. Como se ha visto, es habitual para los algoritmo de *tracking* perder el objetivo en ciertos momentos. Alguno, como los de análisis de secuencias de larga duración sobre todo, están preparados para recuperar la posición del objetivo automáticamente.

- **Refinamiento del template**. En la sección 3.4 se mencionan desarrollos que finalmente no fueron incluidos con el fin de preservar la sencillez del algoritmo. Uno de estos desarrollos es un sistema para afrontar los cambios potenciales de perspectiva del objetivo. El desarrollo de este proceso junto con los antes mencionados mejoraría notablemente el rendimiento del algoritmo.

Bibliografía

[1] A. D. Arnold W. M. Smeulders, Rita Cucchiara, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. 5, 6

[2] W. H. Xi Li, "A survey of appereance models in visual object tracking," *ACM Trans. Intell. Syst. Technol. 4*, 2013. 8, 15

[3] K. M. Y. Yannick Verdie, "Tilde: A temporally invariant learned detector," *Computer Research Repository*, 2014. 8, 13, 14, 18, 21, 24

[4] T. Lindeberg, *Scale-Space in Computer Vision*. Citeseer, 1994. 10

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004. 11, 24

[6] V. N. B. A. Rodriguéz, "Maximum margin correlation filter: A new approach for localization and classification," *IEEE Transactions on Image Processing*, pp. 631–643, 2013. 15

[7] J. B. A. González, R. Martín-Nieto, "Single object long-term tracker for smart control of a ptz camera," 2013. 17

[8] C. Harris and M. Stephens, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, p. 50, Citeseer, 1988. 17

[9] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994. 17

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, pp. 2564–2571, IEEE, 2011. 17

[11] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace/Defense Sensing, Simulation, and Controls*, pp. 95–102, International Society for Optics and Photonics, 2001. 24

[12] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 2, pp. 319–336, 2009. 32

[13] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, "The visual object tracking vot2015 challenge results," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 1–23, 2015. 33

Apéndice A

Scripts

En este apéndice se expondrá el código desarrollado para la implementación del algoritmo. Los detalles sobre su funcionamiento se encuentran en el capitulo 3.

A.1. Programa principal (`main.m`)

Algoritmo A.1 Opciones e inicialización

```
%% Opciones %%

num_PTILDE=20;          %Numero max de puntos tilde
umbral_TILDE=Inf;       %Umbral para la selección de puntos
radio=10;               %Radio del circulo para la correspondencia
threshold=1.5;          %Umbral para la correspondencia
check=true;             %Activar/Desactiar Checkback en la correspondencia
inc_AoI=0.8;            %pocentaje de b que se le anyade al AoI
f_start=1;              %cuadro en el que comenzar a anlizar la secuencua
bb_manual=0;            %Activar/desactivar lectura manual del bb

%% Inicialización %%
%Cargar rutas necesarias y secuencias
addpath(genpath('../TILDE-1.0.4/matlab'));
sequences=dir('../VOT2016');

%Iniciar procesamiento de cada secuencia
for mx=1:numel(sequences)
        %Cargar secuencia y su obtener numero de cuadros
        in_path=['../VOT2016/' sequences(mx).name '/'];
        frames=dir([in_path '*.jpg']);

        %Leer primer cuadro y obetener su tamaño
        I1=imread([in_path frames(f_start).name]);
        [Htot,Wtot,~]=size(I1);

        %Inicializar variables para evaluacion
        accuracy_rate=zeros(1,numel(frames)-1);     %Precision
        fails=0;                                    %Numero de fallos
        fail_idx=[];                                %Indice del fallo
        proc_time=zeros(1,numel(frames)-1);         %Tiempo de procesamiento

        %Abrir y leer el Groundtruth hasta llegar al comienzo elegido
        fid=fopen([in_path 'groundtruth.txt'],'r');
        for lin=1:f_start
                data=fscanf(fid,'%s',1);
        end

        %Determinar la posicion inicial del objetivo, bb inicial
        if bb_manual==0&&exist([in_path 'groundtruth.txt'],'file')==2
                %Desde el groundtruth
                data=strsplit(data, ',');
                [bb]=getSolution(data);
        else
                %Manualmente
                imshow(I1);         [xM,yM]=ginput(2);
                bb.x1 = xM(1); bb.x2 = xM(2);bb.y1 = yM(1); bb.y2 = yM(2);
                close all;
        end

        %Definir dimensiones de bb y del AoI
        [nCbb,nRbb,WAoI,HAoI]=defineSizes(bb,inc_AoI);
```

Algoritmo A.2 Detección de puntos de interés

```
        for f=f_start:numel(frames)-1
                tic;     %Iniciar cronometro

                %% Extracción de caracteristicas
                %Direnciar entre la primera y las sucesivas iteraciones
                if f==f_start
                        %Primera iteracion: cargar I2 y features 1 y 2
                        I2=imread([in_path,frames(f+1).name]);
                        features1=cargaPTILDE(frames(f).name,in_path,I1);
                        features2=cargaPTILDE(frames(f+1).name,in_path,I2);
                else
                        %Siguientes iteraciones: features1 e I1 son los de la
                            iteracion
                        % anterior, cargar/calcular features2 e I2
                        I1=I2;
                        features1=features2;
                        I2=imread([in_path,frames(f+1).name]);
                        features2=cargaPTILDE(frames(f+1).name,in_path,I2);
                end

                %Actualizar la posicion del AoI
                [AoI]=updateAoI(bb,inc_AoI);

                %% Constricción espacial
                [features1AoI,features2AoI]=selectTILDEPoints(features1,features2,
                    AoI,umbral_TILDE,num_PTILDE);

                %% Descripcion con SIFT
                [f1,d1]=describePoints(I1,features1AoI);
                [f2,d2]=describePoints(I2,features2AoI);
```

Algoritmo A.3 Correspondencia y constricción espacial

```
                %% Correspondencia con SIFT
                [my_matches,my_scores]=myMatch(d1,d2,f1,f2,radio,threshold,check);

                %seleccionar correspondencias positivas
                f1Match=f1(1:2,my_matches(1,:));
                f2Match=f2(1:2,my_matches(2,:));

                %% Definicion del área de busqueda
                [template,IAB,AB]=getTemplates(I1,I2,bb,AoI,f2Match);

                %% Busqueda
                [cx,cy,bb,~]=getBoundingBox(template,bb,AB,IAB,I2);
                BBs(f)=bb;
```

Algoritmo A.4 Evaluación

```
                %% Evaluar resultados
                %Leer bb correspondiete del Grountruh
                data=fscanf(fid,'%s',1);
                data=strsplit(data,',');

                %Calcular precisión
                [accuracy_rate(f)]=getAccuracy(data,bb,Htot,Wtot,0);

                %Detectar y recuperar fallos
                if accuracy_rate(f)==0
                        %Contabilizar fallos
                        fails=fails+1;
                        fail_idx=[fail_idx f];

                        %Recuperar de bb de Grountruth
                        [bb]=getSolution(data);
                        [nCbb,nRbb,WAoI,HAoI]=defineSizes(bb,inc_AoI);
                end

                %% Representar desarrollo de la secuencia
                plotSequence(I2,template,IAB,f2Match,bb,AoI,AB,inc_AoI,cx,cy,f);

                %Parar cronometro
                proc_time(f)=toc;

        end
        %Calcular precision globar y cerrar el archivo Groundtruth
        accuracy=sum(accuracy_rate)/f;
        fclose(fid);

        %Guardar los datos de evaluación
        save([in_path 'test.mat'],'accuracy_rate','BBs','accuracy','fails','
            fail_idx','proc_time');
end
```

Apéndice B

Datos sobre la instalación

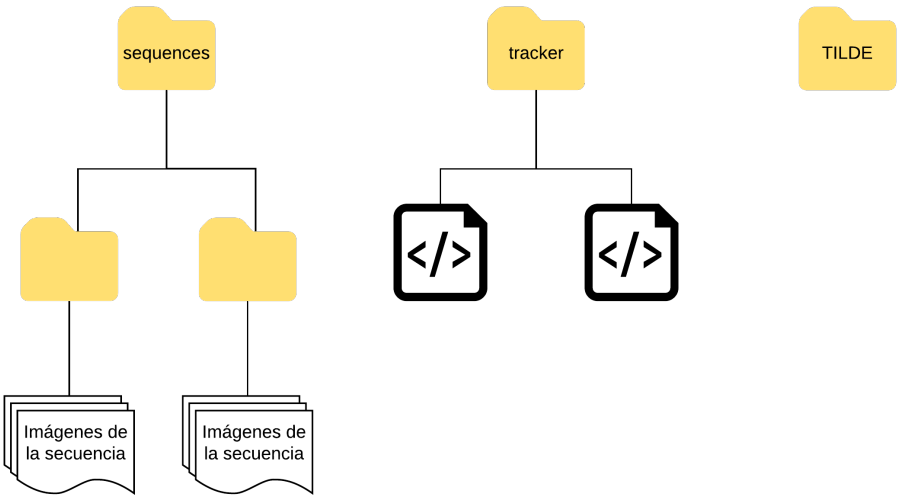

Figura B.1: Estructura de la instalación